protocol it is necessary to implement the port in the software. This is achieved using the micro-controller of the image capture controller 19. The dsp communicates with the UART 22 of the controller 19 using standard asynchronous serial communications 23. The controller 19 takes the commands and data and passes them onto the image sensor 18. There are also commands from the dsp 17 to change settings and modes on the controller 19.

In an alternative arrangement the I^2C communications can be carried out directly from the dsp 17. This can be either a hardware I^2C port or a software implementation.

A sensor driver software module controls communications and interactions with the image capture controller 19 and the image sensor 18. The sensor driver software module has buffers for storing messages to go to the controller 19 and routines for processing messages returned. It also implements the sending and receiving of the video memory token.

The dsp code includes an image processing controller which acts as a wrapper for the image processing routines. Instead of calling the image processing routines directly when a new image is available (token received from controller 19) the image processing controller is called. The image processing controller is responsible for updating any information needed by the image processing routines, updating information that is needed for the image header, starting and stopping watchdog supervisory routines and sending the video memory token back to the image capture controller 19.

The storing of compressed images is an important feature of the DVC.

In most imaging systems analogue video or digital, the camera simply sends out the image information as it is collected. This means that communications must be of sufficient bandwidth to cope with full rate real time images. It also implies that there is some means of buffering the imaging at the other end of the chain for review.

As mentioned previously, the DVC includes memory set aside to store several tens of images. This means that the DVC can be capturing images without sending any out and have past history available on demand should it be needed. Because there is not a constant stream of imaging information coming out, a low bandwidth digital network can be used without fear of it being overloaded by imaging.

Unless adaptive, most lossy coders vary in performance with the complexity of the scene. All coder parameters being equal, a very complex scene (lots of detail) takes up more room than a less complex one. This means that the images stored in the DVC are assumed to be of variable length.

The requirements for image storage are:-

- Store as many images as is possible, delete them only when necessary.
- Be able to efficiently search the image buffer.
- Be able to send any of the stored images in any particular order.
- Images to be contiguous in memory to allow the use of DMA.
- Interface to put images in and read them out to be simple.

Figure 3 shows the organisation of the memory used for the image buffer. There is an array 28 of words which is statically declared. Space in array 28 is dynamically allocated to images as they are created. The images are indexed by an array of ponters 29 to image structures. The set of valid images are bounded by the oldest 0 to newest N images. These are indicated by pointers into the image pointer array 29. Both the image data array 28 and the image pointer array 29 are circular in nature. The wrapping around is handled by image buffer functions. Images cannot be allocated over the end of the image data array 28. If they run off the end, they are moved to the front and the free space 30 at the end is left vacant.

The images in the image data array are stored compressed in a large, statically allocated array of integers (words). There is a header on the front of every image with the following format:-

|Image Status |Size |Compression settings (3 words) |
Request number |Motion Status |Time Stamp| Data

Each of the header entries is one dsp word or 24-bits, except for the time stamp and compression settings. The image status is a state variable for a state machine that describes the behaviour of the image.

Figure 4 shows the transitions of the image states and the functions that cause them. There are two sets of functions that modify the image status. The functions for building the image and the functions for sending the image.